
Wintersemester 2016/2017

Bachelor- Praktikum Hinweise für Teilnehmer

Bachelor-Praktikum

Die wesentlichen Elemente
des
Bachelor-Praktikums
am
Fachbereich Informatik
der
Technischen Universität
Darmstadt

Dr. Michael Eichberg (Redaktion)
eichberg@informatik.tu-darmstadt.de
(Gesamtverantwortung)
Michael Reif
reif@st.informatik.tu-darmstadt.de
(Organisation und Ablauf)
Johannes Lerch
lerch@st.informatik.tu-darmstadt.de
(Organisation und Ablauf)

Vorwort

Das folgende Dokument beschreibt die wichtigsten Aspekte rund um das Bachelor-Praktikum des Fachbereichs Informatik der Technischen Universität Darmstadt.

Es dokumentiert insbesondere auch Erkenntnisse sowie Antworten auf Fragen, die im Laufe der Jahre aufgekommen sind. Das Dokument ist deswegen ein lebendes Dokument, das der ständigen Weiterentwicklung unterliegt.

Struktur

Das Bachelor-Praktikum des Fachbereichs Informatik der Technischen Universität Darmstadt:

- Bachelor-Praktikum (9CP)

Bachelor-Praktikum

Ein Praktikum des Fachbereichs Informatik, bei dem über ein Zeitraum von **6 Monaten** eine umfangreiche(re) Aufgabenstellung in einem Team von vier Studierenden bearbeitet wird. Die Aufgabenstellungen werden von den Fachgebieten der Technischen Universität Darmstadt gestellt. Die Organisation und fachliche Betreuung erfolgt durch den Fachbereich Informatik. Das Bachelor-Praktikum ist verpflichtend für Bachelor-Studierende des Studienganges Bsc Informatik und Psychologie in IT.

Projektbegleitung

Die Projektbegleitung ist eine integrierte Lehrveranstaltung zur Begleitung des Bachelor-Praktikums.

Die Projektbegleitung ergänzt die Veranstaltung (Einführung in) Software Engineering und dient der Vermittlung / Wiederholung von Kenntnissen, die notwendig sind für die systematische Softwareentwicklung. Insbesondere wird auf die Darstellung von hilfreichen Werkzeugen zur Durchführung von Software Engineering Projekten eingegangen. Einen weiteren Schwerpunkt bildet die **Erstellung des Qualitätssicherungsdokuments**.

Darüber hinaus werden im Rahmen der Projektbegleitung die **Projektthemen durch die Studierendengruppen präsentiert**.

Ablauf

Bachelor-Praktikum

- **Start - “jetzt”**

D.h. der Teamleiter wird sich (sehr zeitnahe) bei seinen Gruppen melden, um ein erstes Treffen zu vereinbaren.

Gemeinsam mit dem Teamleiter wird dann ein **Treffen mit dem Auftraggeber vereinbart**.

(im Normalfall im Laufe der nächsten Woche; spätestens jedoch übernächste Woche)

- **Ende - im SoSe 30.9./ im WiSe 31.3.**

Abgabe aller Dokumente und Beendigung der Implementierung

(Dieser Termin ist strikt und unter keinen Umständen verhandelbar.)

Projektbegleitung

(Wintersemester 16-17)

- **28.10.16** Intro - Organisatorischer Ablauf des BP
- **04.11.16** Vorstellung HDA, User Stories, QS-Dokument
- **11.11.16 - 25.11.16** Tools
- *Mitte November* Vorabgabe des QS-Dokuments über Teamleiter
- **02.12.16** Besprechung Vorabgabe QS-Dokument
- **09.12.16 - 23.12.16** entfällt für HDA Vortragstraining
- **13.01.16 - 02.02.17** Projektpräsentationen
(4 Termine; vollständige Anwesenheit von allen an allen Terminen wird erwartet)
- *Mitte/Ende Januar* Abgabe des QS-Dokuments
- **10.02.17** Besprechung QS-Dokumente, Abschluss der Projektbegleitung
- **31.03.17** Projektende

Abgaben

Im Folgenden wird beschrieben welche Artefakte am Ende des Projektes abzugeben sind. Die Art der Dokumente, die abzugeben sind, richtet sich nach dem gewählten Softwareentwicklungsprozess und ist im Folgenden beschrieben.

Das Dokumentenformat muss in allen Fällen PDF sein. Über die im Folgenden genannten Artefakte hinaus sind keine weiteren Dokumente (insbesondere kein Sourcecode oder vergleichbares) bei der Projektbegleitung abzugeben.



Prozessunabhängig

Am Ende des Projektes ist immer ein **Projekttagebuch** abzugeben. Dies gilt auch dann, wenn das Projekttagebuch leer ist. In diesem Fall sollte das Tagebuch lediglich den Eintrag "keine Einträge" enthalten. (*Eine Abgabe ist deswegen gefordert, um unterscheiden zu können zwischen (a) "die Abgabe wurde vergessen" und (b) "ein leeres Projekttagebuch" wurde abgegeben.*)

Agiler Softwareentwicklungsprozess

Sollte der agile Softwareentwicklungsprozess gewählt werden, dann sind folgende Artefakte abzugeben.

- **Qualitätssicherungsdokument**
- **Anhang zum Qualitätssicherungsdokument**
- **User Stories bzw. erfasste Anforderungen**
- (ggf. Anhang zu den User Stories)

“Unified Process”

In diesem Fall ist ein **Pflichtenheft** zu erstellen, dass vom Auftraggeber abgenommen werden muss und alle wesentlichen Abschnitte aufweisen muss.

Für die weiteren Details nehmen Sie bitte Kontakt zur Projektbegleitung auf.

Projekttagbuch

Das Projekttagbuch dokumentiert zwei wesentliche Aspekte: (1) die Wahl der Softwarelizenz bzw. die Regelung, die zwischen Auftraggeber und Projektgruppe bezüglich der Nutzung der Software getroffen wurde und (2) projektgefährdende Ereignisse. Im Regelfall enthält das Projekttagbuch in Hinblick auf (2) keine Einträge.

Beispiele für projektgefährdende Ereignisse sind:

- ernsthafte Erkrankungen eines oder mehrerer Teammitglieder/des Teamleiters/des Auftraggebers
- der Auftraggeber hat Zusagen (zum Beispiel in Hinblick auf das zur Verfügung stellen von Hardware/Software oder Spezifikationen) nicht eingehalten
- der Auftraggeber (d.h. der Ansprechpartner) wechselt
(In diesem Fall nehmen Sie bitte auch umgehend mit der Projektbegleitung Kontakt auf.)

- die Aufgabenstellung wird während der Projektphase grundlegend geändert
(In diesem Fall nehmen Sie bitte auch umgehend mit der Projektbegleitung Kontakt auf.)
- ein Teammitglied verlässt die Gruppe
- ein Teammitglied hält sich (wiederholt) nicht an Absprachen bzw. Zusagen

Qualitätssicherungsdokument

Überblick

Das Ziel des Qualitätssicherungsdokuments ist es zu beschreiben, wie die Qualität des zu erstellenden Produktes im Rahmen des gewählten Projektes gesichert wird. Dies umfasst sowohl die **(I) Beschreibung des Prozesses der Qualitätssicherung als auch (II) der Ziele und (III) der gewählten Maßnahmen.**

Der mit Projektende abzugebende Anhang belegt, dass die Qualitätssicherung wie beschrieben durchgeführt wurde.

Dabei ist ausgehend von den – für das Projekt relevanten und im Idealfall mit dem Auftraggeber abgestimmten – Qualitätszielen zu beschreiben, wie die entsprechenden Qualitätssicherungsmaßnahmen aussehen. Es ist darauf zu achten, dass **alle im Dokument beschriebenen Prozesse, Ziele und Maßnahmen im Kontext des konkreten Projektes erreichbar und durchführbar sind.**

Darüber hinaus muss das Dokument den Qualitätssicherungsprozess beschreiben. D.h. **wann** wird **welche Maßnahme durch wen** durchgeführt und wie wird auf gefundene Qualitätsprobleme **reagiert**.

Ist eine Erreichung – der durch den Auftraggeber (explizit oder implizit) vorgegebenen Qualitätsziele – im Rahmen des Projekts nicht (mehr) möglich, so ist zu begründen warum dies nicht mehr möglich ist und warum davon am Anfang davon ausgegangen wurde, dass die Ziele erreichbar und die Maßnahmen durchführbar sind. Weiterhin ist darzulegen, welche alternativen Qualitätsziele verfolgt wurden und welche Maßnahmen mit welchem Prozess durchgeführt wurden.

Eine mögliche Maßnahme, um zum Beispiel hohe Codequalität zu erreichen, ist die regelmäßige, strukturierte Durchführung von Code Reviews oder der regelmäßige Einsatz von Tools zur statischen/dynamischen Analyse.

Eine mögliche Vorlage

(I) Beschreibung des projektspezifischen Qualitätsziels & kurze Begründung:

Im Rahmen dieses Projektes hat die Sicherstellung der ...höchste Priorität, da

(II) Beschreibung der Maßnahmen, die durchgeführt werden, um das Erreichen des Ziels sicherzustellen:

Um das Qualitätsziel zu erreichen, führen wir regelmäßig Tests/Studien/Code Reviews/Schulungen/... durch. [Aussagen, die die Maßnahme genau beschreiben.] Weiterhin setzen wir folgende Werkzeuge ... ein, die

(III) Beschreibung des Prozesses:

Die Maßnahme wird alle ... durchgeführt. Die Ergebnisse werden... diskutiert und einem Entwickler zum Lösen zugewiesen. Für geänderten Code wird der selbe Qualitätssicherungsprozess angewandt wie für neuen.

Allgemeine Hinweise

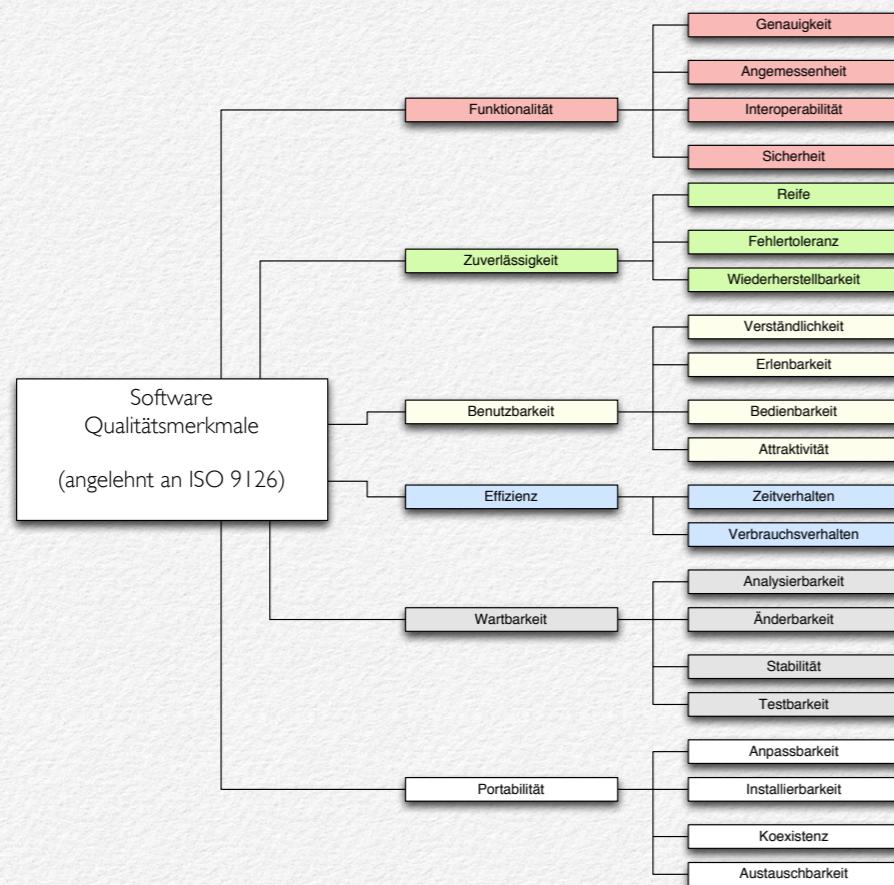
- Es handelt sich um ein Dokument im TU Darmstadt Design mit max. 10 Seiten (ohne Anhang); **die von der Projektbegleitung zur Verfügung gestellte Vorlage ist zu verwenden (auch für die Vorabgabe).**

Auf dem Deckblatt sind zu vermerken:

- Das Projektthema/der Titel des Projekts
- Die vollständigen Namen der Teammitglieder
- Kontakt (EMail)
- Der vollständige Name des Teamleiters
- Name der Auftraggeber(in)/des Auftraggebers (inkl. Fachgebiet/Fachbereich)
- Gruppennummer
- Ein Qualitätssicherungsziel, dass von Seiten der Projektbegleitung *für die Vorabgabe* gesetzt ist – und dass alle Gruppen im Qualitätssicherungsdokument beschreiben müssen – ist die Sicherung der Qualität des Codes im weiteren Sinn. Hierbei ist zu beachten, dass auch dieses Ziel auf das konkrete Projekt angepasst werden muss. Für das finale Dokument kann das Ziel ggf. gestrichen werden.

- Bitte beachten Sie, dass – sofern der Auftraggeber nichts anderes wünscht – das Qualitätssicherungsdokument unter keinen Umständen eine Einführung (in welcher Hinsicht auch immer) in Qualitätssicherung enthält.
Sollte der Auftraggeber dies explizit wünschen, so ist dies auch explizit auf dem Dokument zu vermerken.
- Vergessen Sie nicht die Qualität des Qualitätssicherungsdokumentes zu sichern!
Stellen Sie insbesondere sicher, dass die folgenden Fragen in Ihrem Dokument beantwortet sind:
 - Ergibt sich aus der Projektbeschreibung, warum die dargestellten Qualitätsziele, die essentiellen/zentralen QS-Ziele des Auftraggebers sind?
 - Warum sind dies die zentralen Ziele?
 - Welche Maßnahmen werden durchgeführt?
 - Wie dienen die Maßnahmen dem Erreichen der gegebenen Ziele?
 - Sind die Maßnahmen dafür geeignet die Ziele zu erreichen?
 - Durch wen werden die Maßnahmen durchgeführt und wie wird auf Probleme bis wann reagiert?
 - Ist die Häufigkeit, mit der die Maßnahmen durchgeführt wird angemessen?
 - Ist die Durchführbarkeit der Maßnahmen sichergestellt?

Qualitätsmerkmale



Korrektheit

Grad der Konsistenz zwischen Spezifikation und Programm bzw. als Grad der Erfüllung der Benutzererwartung durch ein Programm (d.h. ohne Spezifikation ist keine Korrektheit nachweisbar).

Vollständigkeit

Alle geforderten Funktionen sind realisiert.

Sicherheit (eng. Safety)

Eigenschaft eines Systems weder Menschen, noch Sachen oder die Umwelt zu gefährden.

Datensicherheit (eng. Security)

Eigenschaft eines Systems Informationsverluste und unbefugten Datenzugriff zu verhindern.

Zuverlässigkeit

Die Wahrscheinlichkeit des ausfallfreien Betriebs der (in diesem Kontext) Software über einen bestimmten Zeitraum bei einer definierten Betriebsweise.

Verfügbarkeit

Eigenschaft zu einem gegebenen Zeitpunkt funktionstüchtig zu sein.

Robustheit

Im Wesentlichen eine Eigenschaft der Spezifikation. Resultiert im Wesentlichen aus der korrekten Umsetzung einer Spezifikation, die auch ungewöhnliche Betriebssituationen erfasst.

Beschreibung eines Qualitätszieles

Beispiel

Projektspezifi-
sche Motivati-
on des Projekt-
zieles

“Scope”/ Um-
fang/ Ziel

Qualitätsziel: Sichere Webanwendung

„Im Rahmen des Projektes ... wird eine Webanwendung entwickelt, auf die über das Internet zugegriffen wird. Da diese Anwendung ... personenbezogene Daten verarbeitet und potentiellen Angriffen ausgesetzt ist, ist ein wesentliches Qualitätsziel, dass die Anwendung keine Sicherheitslücken aufweist über die Angreifer Daten anderer Benutzer abgreifen können.

Im Rahmen dieses Projektes können wir jedoch nur gewährleisten, dass die Webanwendung keine „Standardlücken“ wie zum Beispiel SQL Injection aufweist. Um dieses Ziel zu erreichen, setzen wir die folgenden Tools: ... ein. Darüber hinaus wurde ein Entwickler benannt, der sich maßgeblich um das Thema „Sicherheit in Webanwendungen“ kümmert und ...

Die automatisierte Analyse des Codes der Webanwendung erfolgt im Rahmen des regelmäßigen „Nightly Builds“. Sollte ein Problem gefunden werden, so geht eine Mail an alle Entwickler und im Rahmen des nächsten (gruppeninternen) Meetings wird dann ein Entwickler bestimmt, der den Fehler beseitigt.

Wann und
durch wen?

Wie wird da-
rauf reagiert?

Projektspezi-
sche Motivati-
on des Projekt-
zieles

“Scope”/ Um-
fang/ Ziel

Qualitätsziel: Benutzbarkeit

Im Rahmen des Projektes, wird eine Software.

Wann und
durch wen?

Wie wird da-
rauf reagiert?

Typische Probleme

(Alle im Folgenden wiedergegeben Ausschnitte entstammen (Vorabgaben von) Qualitätssicherungsdokumenten der vergangen Jahre.)

Aussagen aus QS Dokumenten

Außerdem soll eine Evaluation die Benutzerfreundlichkeit der Anwendung gewährleisten.

Die Rechtschreibung – insbesondere in Hinblick auf die Anwendung der Kommaregeln – ist sehr schlecht.

Tipp:

- Komm wir essen Opa (Satzzeichen retten leben)
- Eats(,) Shoots & Leaves

Einzelne Sätze machen – schon für sich alleine genommen – keinen Sinn.

Außerdem soll eine Evaluation die Benutzerfreundlichkeit der Anwendung gewährleisten.

Durch eine Evaluation kann man die Benutzerfreundlichkeit ermitteln, aber nicht gewährleisten!

Bindestriche werden “wahllos” platziert.

Qualitäts-Faktoren

Umgangssprachlich

„Hier beißt sich also die Katze in gewissem Sinne in den eigenen Schwanz.“

Die Struktur des Dokumentes hält sich nicht an die etablierten Regeln. Insbesondere gibt es Kapitel mit nur einem Unterkapitel statt mindestens zwei Unterkapiteln.

Tipp: Nehmen Sie sich ein (ggf. elektronisches) Buch eines renommierten Verlages und strukturieren Sie das Dokument in vergleichbarer Weise.

Deutsch und Englisch werden wild gemischt.

Im Weiteren werden folgende Qualitätsfaktoren [...]:

Correctness

Robustness Extendibility

Es wird nicht präzise beschrieben, was getan wird. Statt dessen wird beschrieben, was möglicherweise getan werden könnte.

Um dies zu beschleunigen, wollen wir eine Dokumentation über die Schnittstellen unserer Software bereitstellen.

Der Qualitätssicherungsprozess ist nicht zu Ende gedacht bzw. nicht ausreichend beschrieben.

Am Ende des Projektes führen eine Benutzerstudie durch, um [...] zu erreichen.

Dinge, die am Ende des Projektes gemacht werden, können faktisch keine wesentliche Auswirkung mehr auf das Produkt haben und somit der Erreichung eines Qualitätsziels nicht mehr dienen.

Tipp: Geben Sie immer (relativ) präzise an, wann Sie eine Maßnahme durchführen werden und überlegen Sie sich genau, ob der Zeitraum, der nach der Durchführung der Maßnahme noch zur Verfügung steht, ausreicht, um ggf. das Ziel noch (teilweise) zu erreichen.

Es werden bei weiten zu viele Qualitätsziele und Maßnahmen beschrieben und verfolgt.

Tipp: Fragen Sie sich genau was im Rahmen dieses Projektes am Wichtigsten ist, wie viel Sie zeitlich leisten können und an welchen Stellen die größten Risiken liegen bzgl. der Nichtereichung des Projektziels.

Der Abstraktionsgrad innerhalb eines Abschnitts springt wild hin und her.

Es werden Qualitätsziele, die für das Projekt nicht relevant sind, und Maßnahmen, die nicht durchgeführt werden, beschrieben.

Zu beschreiben, was nicht getan wird, macht fast ausschließlich nur dann Sinn, wenn Sie den Umfang einer Maßnahme begrenzen wollen auf ein – für das Projekt – sinnvolles Maß.

“Jedes” eingesetzte Werkzeug wird als Qualitätssicherungswerkzeug betrachtet.

Als integrierte Entwicklungsumgebung (IDE) wird Netbeans verwendet, wodurch Syntaxfehler vermieden werden.

„QS-Werkzeuge[...] Skype [Sky] ist eine kostenlose Voice Over IP (VOIP) Software, welche zusätzlich noch die Funktionen zu Instant Messaging, Dateiübertragung und Videotelefonie bietet. Im Rahmen des Projektes wurde ein Chatraum in Skype eingerichtet, in welchem sich täglich über das Projekt ausgetauscht wurde. Skype unterstützte die Teamarbeit.“

Integrierte Entwicklungsumgebungen, Versionskontrollsysteme, Projektmanagementsoftware und Chatclients dienen primär/ ausschließlich der Produktivität bzw. der Steuerung des Projektes jedoch nicht der eigentlichen Produktqualität.

“Blah, Blah, Blah”

Unser Auftraggeber ist zufrieden, wenn unser Produkt am Ende eines Sprints die Akzeptanzkriterien der mit ihm im Gespräch entwickelten User Stories erfüllt. Wenn unser Auftraggeber zufrieden ist, sind wir zufrieden.
[...]

Tipp: Bedenken Sie immer die Zielgruppe, für die Sie ein Dokument erstellen. In diesem Fall dürfte es fast immer (nur) die Pro-

jektbegleitung sein. Schreiben Sie also nichts, was Personen, die sich seit vielen Jahren mit Fragen der Softwaretechnik beschäftigen, auf jeden Fall wissen. Sollte der Auftraggeber Interesse an dem Qualitätssicherungsdokument zeigen und dieses (weiter)verwenden wollen, dann benennen Sie die Zielgruppen bitte am Anfang des Dokumentes.

Der Projektbezug ist nicht gegeben und es handelt sich um eine allgemeingültige Feststellung.

„Inversion des Kontrollflusses: Durch den Einsatz des Frameworks wird erreicht, dass sich der Entwickler nicht mehr um den Kontrollfluss der Web-Anwendung kümmern muss. Das erledigt das Framework für ihn.“

Statt einem Qualitätsziel wird eine (nicht-)funktionale Anforderung beschrieben.

Schwammige Aussagen

Um möglichst fehlerfreien Code zu erreichen, werden wir umfangreiche Tests zusätzlich zum Code entwickeln.

Hier stellt sich unter anderem die Frage nach dem Umfang von “umfangreichen Tests”? Ähnliche Aussagen finden sich häufig auch in Hinblick auf die Dokumentation des Codes.

Tipp: Es ist häufig hilfreicher/einfacher den Umfang einer Testsuite über die (Kern-)Komponenten bzw. über die umzusetzenden Features zu beschreiben.

Der Anhang des Qualitätssicherungsdocuments

Der Anhang muss dokumentieren, dass die beschriebenen Qualitätsmaßnahmen und Prozesse auch durchgeführt wurden. Es ist hierbei insbesondere darauf zu achten, dass erkenntlich ist, dass der Prozess eingehalten wurde (d.h. wann und wie häufig etwas getan wurde) und auch, dass die Maßnahmen im beschriebenen Umfang durchgeführt wurden.

Im Folgenden stellen wir für typische Maßnahmen bzw. Ziele dar, wie diese ggf. zu belegen sind. Lesen Sie auf jeden Fall alle Abschnitte, da viele Dinge ggf. auf Ihre Maßnahmen und Ziele übertragbar sind.

Benutzerstudie

Im Anhang muss dokumentiert sein, wann diese Studie(n) von wem und mit welchen Probanden durchgeführt wurde.

Darüber hinaus sind die ausgefüllten Fragebögen und auch die entsprechenden aggregierten Ergebnisse anzuhängen.

Wurden die Probanden beobachtet, dann sind die entsprechenden Protokolle anzuhängen.

Insbesondere ist kurz zu dokumentieren, welche Ergebnisse aus der Benutzerstudie abgeleitet wurden und welche Konse-

quenzen gezogen wurden (z.B. durch Referenz auf zeitlich nachfolgende User Stories).

Dokumentation des Quellcodes

Ist eine Maßnahme, die versprochen wurde, dass der Code dokumentiert wurde, so ist hier ein Auszug des Codes abzugeben. Der Auszug muss aus mindestens drei verschiedenen Klassen, Headerfiles oder etwas vergleichbaren Bestehen. Die Dateien müssen von der Anzahl der Zeilen Code her repräsentativ für das Projekt sein. Die gewählten Dateien müssen weiterhin von herausgehobener Bedeutung für das Projekt sein und dies muss entweder aus der Dokumentation hervorgehen oder gesondert kurz dargelegt werden. Der Code ist direkt in den Anhang zu integrieren.

Automatisierte Tests

Sollte das QS Dokument angeben, dass die Software automatisiert getestet wurde, so ist die vollständige Liste der Tests abzugeben und es ist zu belegen welche Teile des Codes getestet wurden.

Dies kann insbesondere dadurch geschehen, dass ein Auszug eines Codeabdeckungstools angehängt wird; dargestellt auf Klassen-/Dateiebene. Bitte geben Sie auf jeden Fall für die Tests auch an auf welche Anforderungen/User Stories/Funktionalität sich die Tests beziehen. Geht aus dem Namen des Tests hervor, worauf sich selbiger bezieht, so ist dies ausreichend.

Modularität bzw. Erweiterbarkeit

Um zu belegen, dass das Ziel erreicht wurde sind die zentralen Schnittstellen inkl. der unmittelbaren Abhängigkeiten zu dokumentieren.

Darüber hinaus sind die Abhängigkeiten zwischen den Klassen/Modulen/Packages/Maven Artefakten/... genau zu dokumentieren.

Abzugeben ist die Dokumentation des Standes der Erweiterbarkeit/Modularisierung am Ende des Projektes zuzüglich einer kurzen Übersicht, wann die Maßnahmen – Analyse der Struktur, Definition und Besprechung der Schnittstellen, etc. – durchgeführt wurden.

Code Reviews

Diesbezüglich ist eine Checkliste anzuhängen – falls diese nicht bereits im Hauptteil dargestellt ist – aus der genau hervorgeht, was alles geprüft wird und wann die Prüfung als erfolgreich gilt.

Es sind auch alle ausgefüllten Checklisten anzuhängen. Aus diesen muss jeweils hervorgehen, wann die Prüfung durch wen erfolgt ist und welche Teile einem Review unterzogen wurden. Etwaige gezogene Konsequenzen sind zu dokumentieren (analog zu “Benutzerstudie”).

Manuelle Tests

In diesem Fall ist der Testplan anzugeben – falls er nicht bereits im Hauptteil angegeben wurde. Aus dem Testplan muss hervorgehen, dass das beschriebene Qualitätsziel in vollem Umfang sichergestellt werden kann. Ein Testplan muss immer mind. folgende Informationen beinhalten: die durchzuführenden Schritte und die jeweiligen erwarteten Ergebnisse.

Darüber hinaus sind die Protokolle abzugeben, welche die Ergebnisse der jeweiligen Durchführung und die ggf. gezogenen Konsequenzen angeben. Es ist auch anzugeben, wann die Tests von wem durchgeführt wurden.

Ausgewählte Qualitätsmerkmale bzw.-maßnahmen.

“(Gut/Umfangreich/...) dokumentierter Code”

“Documentation costs much time, usually more than actual implementation. That’s the reason why people try to avoid documentation.”

(From: How Do Professional Developers Comprehend Software?; Tobias Boehm, Rebecca Tiarks, Rainer Koschke, Walid Maalej; ICSE 2012; IEEE)

Sollten Sie dieses Qualitätsziel gewählte haben, dann belegen Sie die gute/ausreichende Dokumentation Ihrer Software dadurch, dass Sie uns die generierte Dokumentation (HTML Seiten, PDF, ...) zur Verfügung stellen. Im QS Dokument ist dann ggf. auf das/die PDFs bzw. die Hauptseite zu verweisen.

“(Gut/...) getesteter Code”

Um zu belegen, dass Sie Ihren Code getestet haben stellen Sie uns folgende Informationen zur Verfügung:

- Eine Analyse der Codeabdeckung (unabhängig davon ob Sie automatisierte oder manuelle Tests durchgeführt haben); spezifizieren Sie genau das zugrunde gelegte Codeabdeckungsmaß und das verwendete Werkzeug. Zum Beispiel mit ECL Emma (bei Java programmen).

- Falls Sie manuelle (GUI-)Tests durchführen, dann ist der Testplan einzureichen.
- Falls Sie automatisierte Tests durchführen, dann ist die vollständige Liste aller (Unit)Tests abzugeben.

In allen Fällen muss erkenntlich sein, wann die Test durch wen und mit welchem Ergebnis durchgeführt wurden. Sollten Tests fehlgeschlagen haben, dann ist zu dokumentieren wann diese behoben wurden.

Werkzeuge

Im Folgenden sind einige wenige Werkzeuge aufgeführt, die im Rahmen einer systematischen Qualitätssicherung eingesetzt werden können.

Java

[Hammurapi](#) (Verletzungen von “Best Practices”)

[PMD](#) (Verletzungen von Konventionen und “Best Practices”)

[Checkstyle](#) (Verletzungen von Konventionen)

[Findbugs](#) (Verletzungen von “Best Practices”)

[JLint](#) (Verletzungen von “Best Practices” - mit Fokus auf nebenläufige Programme) (im Allgemeinen: “x”Lint.)

[ECLEmma](#) (Abdeckung des Codes durch eine Testsuite)

[Java Path Finder](#) (Software Model Checking and much more...)

[JDepend](#) (Abhängigkeitsanalysen)

[Dependency Finder](#) (Abhängigkeitsanalysen)

[Stan](#) (Strukturanalysen)

[Cloc](#) (Metriken)

Sprachunabhängig

[Gerrit](#) (Code Review System)

[Selenium](#) (Primarily, for testing web applications)

[Apache JMeter](#) (Last- und Performancetests)

Artistic Style 2.05 (Codeformatierung für C Dialekte und Java)

Cloud Based

[Shippable](#) (CI+CD - works with Bitbucket and GitHub)

[Travis CI](#) (CI - works with GitHub)

[Wercker](#) (CD - works with)

Benotung

Die Endnote für die Bachelor-Praktikumsgruppe ergibt sich aus der **Bewertung durch den Auftraggeber** und aus der **Bewertung durch die Projektorganisation**. Beide Noten gehen zu 50% in die Endnote ein.

Die Benotung der Projektorganisation stützt sich insbesondere auf die erstellten und abgegebenen Dokumente (basierend auf den Besonderheiten der Projekte).

- Sollte der agile Softwareentwicklungsprozess gewählt worden sein, dann ist dies insbesondere das **Qualitätssicherungsdokument inkl. Anhang**, der belegt, dass die Qualitäts sicherung wie beschrieben durchgeführt wurde. Darüber hinaus wird die vollständige Dokumentation der **Anforderungen** herangezogen.
- Im Falle des “Unified Process” wird insbesondere das **Pflichtenheft** bewertet.

Die Bewertung der Gruppe ist vollständig unabhängig von der Benotung des Teamleiters und der Teamleiter wird auch nicht in die Notengebung der Gruppe eingebunden.

Weiterhin geht der **Vortrag**, der im Rahmen der Vorlesung zu halten ist, in die Endnote ein.

Das **Projekttagebuch** dient - insbesondere für den Fall, dass es im Rahmen der Bearbeitung des Projektes zu Problemen, Verzögerungen oder Streitigkeiten (zwischen den Gruppenmitgliedern/mith dem Auftraggeber) gekommen ist - dazu, zu ermitteln woher die Schwierigkeiten kamen und ob diese (auch) in der Verantwortung der Studierenden lagen. Es dient ggf. als Basis für eine faire Einzelbenotung.

In der Regel wird die Gruppe als Ganzes bewertet. Einzelnoten werden nur im Sonderfall und aufgrund des expliziten Wunsches der Gruppe bzw. bei besonderen Problemen vergeben.

Allgemeine Werkzeuge

Werkzeuge, die häufig im Rahmen des Bachelor-Praktikums genutzt werden können:

- **Git Clients**
 - SourceTree (Git Client)
 - GitHub (Git Client, entwickelt von GitHub)
 - Eclipse + eGit
- **Repository Hosting Services**
 - Bitbucket (auch Private Repositories sind kostenlos)
 - GitHub
- **Projektmanagement**
 - Trello

Agiler Softwareentwicklungsprozess

Im Folgenden wird kurz beschrieben, wie sich ein – von Extreme Programming und Scrum inspiriertes – agiles Softwareprozessmodell im Rahmen des Bachelor-Praktikums umsetzen lässt. Insbesondere geht dieses Dokument darauf ein, an welchen Stellen „Kompromisse“ eingegangen werden müssen, die sich aus der Besonderheit ergeben, dass es sich um eine Lehrveranstaltung handelt.

Der große Vorteil eines agilen Softwareentwicklungsprozesses ist, dass der Auftraggeber in einem kontrollierten Rahmen stets die Möglichkeit hat, das Projekt in die Richtung zu steuern, welche den höchsten Nutzen verspricht.

Grundlagen und Motivation Agiler Softwareentwicklungsprozesse

Das Ziel ist es Software zügig und zielstrebig zu entwickeln in der Gegenwart sich **ständig ändernder Anforderungen**. Um dieses Ziel zu erreichen ist es notwendig **Entwurfsprinzipien**

anzuwenden, die die Software flexibel und anpassbar bzw. wartbar machen. Dazu muss man sich entsprechender **Entwurfsmuster** bewusst sein. Darüber hinaus ist es erforderlich **Vorgehensweisen anzuwenden**, die die nötige Disziplin sicherstellen und den Fortschritt zu ermitteln.

Das wesentliche Ziel ist es, den Auftraggeber zu ermöglichen das Entwicklungsteam in die Richtung zu steuern, welche den **höchsten Geschäftswert** verspricht. Dazu ist es erforderlich, dass man auf Änderungen flexibel reagiert.

Wesentliche Eckpunkte:

- Die regelmäßige und häufige Zusammenarbeit mit dem Kunden
- Die häufige Auslieferung laufender Software, die einen Geschäftsnutzen hat; Fortschritt wird gemessen durch die Funktionalität, die umgesetzt wurde
- Sich selbst organisierende Teams mit motivierten Individuen
- Das Team reflektiert über die Entwicklungsprozesse
- Streben nach technischer Exzellenz und einem guten Entwurf

Geeignete Projekte



Agile Softwareentwicklung eignet sich für Projekte, bei denen die Anforderungen bzw. die Prioritäten der einzelnen Anforderungen an die zu erstellende Software bei Beginn des Projekts nicht vollständig bekannt sind bzw. bei denen es wahrscheinlich ist, dass sich die Anforderungen noch ändern werden.

Dies sind häufig Projekte, bei denen es um die Erstellung von Webanwendungen oder anderweitigen „Geschäftsanwendungen“ geht.

Voraussetzungen auf Seiten des Projektteams

Das Team sollte sich mit den Grundlagen agiler Softwareentwicklungsprozesse auseinandergesetzt haben. Darüber hinaus ist eine stetige Leistungsbereitschaft erforderlich, damit der Auftraggeber seiner Steuerungsfunktion nachkommen kann.

Voraussetzungen auf Seiten des Auftraggebers

Dieses Vorgehensmodell setzt voraus, dass sich der Auftraggeber zu regelmäßigen Treffen – alle 2 bis 3 Wochen – bereit erklärt. D.h. die Wahl dieses Prozessmodells setzt das Einverständnis des Auftraggebers voraus.

Informatikkenntnisse sind auf Seite des Auftraggebers nicht erforderlich - der Auftraggeber bzw. Kunde handelt aus seiner fachlichen Perspektive heraus.

Ablauf während des Praktikums

Start



Im Folgenden wird davon ausgegangen, dass die einzusetzenden Technologien (Programmiersprachen etc.) bereits gesetzt sind – falls dies nicht der Fall ist, dann kann bei Bedarf erst noch eine Explorationsphase vorangestellt werden. Gleiches gilt, falls das Team nicht über notwendige fachliche Kenntnisse verfügt, oder die Vorstellungen des Auftraggebers am Anfang noch sehr vage sind. In diesem Fall bietet es sich an weitere Techniken zur Anforderungsermittlung (z.B. Anwendungsfälle (Use Cases), offene Interviews oder Szenariotechniken) einzusetzen, um ein breites Verständnis für die zu entwickelnde Anwendung zu bekommen.

Im Rahmen der regelmäßigen Treffen mit den Auftraggebern sind die Anforderungen an die zu entwickelnde Software in Form von (neuen) User Stories zu erfassen bzw. gegebene ggf. zu verfeinern. Im Rahmen des ersten Treffens sollte versucht werden, auf einem hohen Abstraktionsniveau alle wesentlichen User Stories zu erfassen. Die User Stories, die im Rahmen der ersten Iteration umgesetzt werden sollen, müssen natürlich im Rahmen der ersten Iteration auch bereits passend herunter gebrochen werden.

Aufbauend auf den erfassten User Stories wird eine Releaseplanung für das gesamte Projekt durchgeführt (zwei bis drei Rele-

ses). Ein Release ist dadurch gekennzeichnet, dass es einen aus fachlicher Sicht sinnvollen Funktionsumfang umfasst. Diese soll dann im Rahmen des zweiten Treffens mit dem Auftraggeber vorgestellt werden.

Während des Projekts



Am Ende/am Beginn jeder Iteration treffen sich der Auftraggeber und das Team und besprechen die Ergebnisse der letzten Iteration. In diesem Rahmen stellt das Team die umgesetzten User Stories vor und der Auftraggeber nimmt diese ggf. ab. Auf Wunsch des Auftraggebers muss das Team in der Lage sein, am Ende jeder Iteration eine lauffähige Version zur Verfügung zu stellen, damit der Auftraggeber diese testen kann, um ggf. die weitere Entwicklungsrichtung zu bestimmen. Sollten neue Anforderungen auftreten, so sind diese unmittelbar als neue User Stories zu erfassen.

Weiterhin wird besprochen welche User Stories im Rahmen der nächsten Iteration umzusetzen sind (Planning Meeting). Es ist darauf zu achten, dass nur so viele User Stories durch den Auftraggeber gezogen werden, wie auch umgesetzt werden können. Falls die Geschwindigkeit des Teams noch nicht bekannt ist bzw. stark schwankt, dann ist es sinnvoll, dass der Auftraggeber den User Stories Prioritäten zuordnet, damit das Team eine genaue Vorstellung davon hat, in welcher Reihenfolge die User Stories umzusetzen sind. Insbesondere werden große User Stories herunter gebrochen, um diese durchführbar zu machen.

Am Ende jeder Iteration stellt das Team eine funktionierende Software dem Auftraggeber vor.

Weiterhin kann es ggf. auch während der Entwicklung notwendig sein zur Ermittlung bzw. Verfeinerung der Anforderungen andere Techniken zur Anforderungsermittlung einzusetzen.

Aufgaben der Praktikumsteilnehmer



Die Hauptaufgabe der Praktikumsteilnehmer ist die iterative und inkrementelle Entwicklung der Software.

Während der Entwicklung muss das Team lückenlos dokumentieren, wer wann welche User Story umgesetzt hat, welcher Aufwand vorher geschätzt wurde (zum Beispiel mit Story Points), und wie hoch der Aufwand (in Std.) war. Darüber hinaus ist die Velocity zu berechnen und als Grundlage für die Planung der nächsten Iteration zu verwenden.

Darüber hinaus sind alle Teammitglieder verpflichtet, die für das Projekt *verbrauchte Realzeit, lückenlos zu erfassen*. Dies umfasst neben der eigentlichen Zeit für die Entwicklung insbesondere auch Aufwände für die Einarbeitung und für Meetings. Im Allgemeinen gilt, dass alle Aufwände, die auch bei einem realen Projekt anfallen, zu erfassen sind.

Sollten während der Entwicklung unvorhergesehene Probleme und Fragen auftreten, die die Erreichung des Projektziels gefährden, so sind diese durch die Teilnehmer (ggf. in Absprache

mit dem Teamleiter) im Projekttagebuch kurz und präzise zu dokumentieren.

Die projektspezifischen Qualitätsziele, Maßnahmen und Prozesse sind in einem Qualitätssicherungsdokument, welches nach ca. der Hälfte der Projektlaufzeit final abzugeben ist, zu beschreiben. Die beschriebenen Maßnahmen und Prozesse werden gelebt und umgesetzt.

Am Ende jeder Iteration muss eine lauffähige Version der Software abgegeben werden und es muss eine kurze Retrospektive geben. Die Leitung der Retrospektive ist im Regelfall Aufgabe des Teamleiters.

Anforderungen an die Auftraggeber

Der Auftraggeber muss sich direkt am Anfang des Projektes hinreichend Zeit nehmen (ggf. mehrere Stunden), um die wesentlichen Anforderungen an das Projekt mit der Gruppe zu diskutieren und auch grob zu priorisieren. Es ist zu diesem Zeitpunkt weder erforderlich noch sinnvoll, zu versuchen, alle Anforderungen detailliert zu diskutieren oder zu ermitteln. Die Anforderungen sind jedoch soweit zu diskutieren, dass das Team eine sehr gute Vorstellung von dem Projekt als Ganzes bekommt. Die Anforderungen können bei Bedarf „jederzeit“ während des Projekts geändert, ergänzt oder gestrichen werden.

Damit dieser Prozess im Rahmen eines Projektes angewendet werden kann, ist es erforderlich, dass der Auftraggeber sich über die Projektdauer zu regelmäßigen Treffen (ideal alle zwei Wochen, maximal alle drei Wochen) verpflichtet. Es ist Aufgabe des Auftraggebers bei der Auswahl der umzusetzenden User Stories für die nächste Iteration(en) mitzuwirken bzw. diese zu bestimmen. Die Auswahl sollte durch den Auftraggeber unter fachlichen Gesichtspunkten erfolgen.

Es obliegt dem Auftraggeber sich regelmäßig mit der Software auseinanderzusetzen, um sicherzustellen, dass die Anwendung den Wünschen entspricht, und um in der Lage zu sein qualitatives Feedback zu der Software zu geben. D.h. vor dem Beginn einer jeden Iteration kann der Auftraggeber ggf. bestehende User Stories anpassen, löschen oder neue hinzufügen.

Der Auftraggeber darf nicht mehr User Stories auswählen, als Zeit zum Umsetzen innerhalb der nächsten Iteration vorhanden ist. Die Information, wie viel Aufwand das Umsetzen einer Story verursacht und wie viel Zeit innerhalb der nächsten Iteration zur Verfügung steht, erhält der Auftraggeber vom Team.

Ist der eigentliche Auftraggeber am Ende einer Iteration verhindert und kann deswegen am Planning Meeting nicht teilnehmen, dann liegt es in der Verantwortung des Auftraggebers, einen qualifizierten Stellvertreter zu benennen. Wenn erforderlich, kann die Dauer einer Iteration auf eine, zwei oder drei Wochen geändert werden. Die Iteration, die sich über die Weih-

nachtsferien erstreckt kann ausnahmsweise bis zu vier Wochen lang sein.



Mit Auftraggeber abzustimmen

Nach/Während des Treffens

Beim Start der Implementierung

Nach Umsetzung

| | |
|------------------------------------|---|
| ID | 2 |
| Name | Login |
| Beschreibung | Als Administrator muss ich mich am System mittels Benutzername und Passwort authentifizieren können, um Änderungen vornehmen zu können. |
| Akzeptanzkriterium | Der Dialog zum Einloggen wird korrekt angezeigt und es ist möglich sich als Administrator zu authentifizieren. Ungültige Eingaben werden ignoriert und normale Nutzer erhalten nicht die Rolle "Administrator". |
| Geschätzter Aufwand (Story Points) | 3 |
| Entwickler | Max Mustermann |
| Umgesetzt | am 2.11.2016 in Iteration 2 |
| Tatsächlicher Aufwand (Std.) | 12 |
| Velocity (Std./Story Point) | 4 |
| Bemerkungen | / |

Beispiel für eine User Story

Die Größe einer User Story muss immer derart sein, dass es möglich ist mehrere in einer Iteration umzusetzen.

“User Stories”

Die wesentlichen Bestandteile einer User Story sind:

1. Die **User Story als solche**, die eine Anforderung des Auftraggebers so beschreibt, dass sich das Team und der Auftraggeber daran erinnert was genau zu tun ist.
User Stories werden – wenn die Story implementiert werden soll – ggf. in Tasks (Aufgaben) weiter untergliedert. Die Aufteilung der User Story in Tasks liegt in der Verantwortung der Gruppe; der Teamleiter ist maximal moderierend tätig.
Ein Format, das öfters für User Stories verwendet wird und welches sich vielfach bewährt hat, ist: „Als <Benutzerrolle> will ich <das Ziel>[, so dass <Grund für das Ziel>]“.
2. Priorität der User Story bzw. ob die User Story gerade in Bearbeitung ist
3. Ein **Akzeptanzkriterium**, dass es erlaubt zu beurteilen, ob die User Story vollständig und korrekt umgesetzt wurde.
4. **Geschätzter Aufwand** - Basis ist die erwartete Komplexität im Vergleich zu anderen User Stories; insbesondere im Vergleich zu bereits implementierten User Stories. Es ist wichtig, dass der geschätzte Aufwand aktuell ist; sollte zwischen der ersten Schätzung des Aufwands und dem Zeitpunkt an dem die User Story umgesetzt werden soll, mehrere Iterationen

liegen, so ist es ggf. sinnvoll die User Story noch einmal zu schätzen!

5. **Tatsächlicher Aufwand in Zeitstunden und die Velocity (Std./Story Point)**, letzteres ist notwendig für die Berechnung der Geschwindigkeit.
Abweichungen zwischen dem geschätzten Aufwand und dem tatsächlichen Aufwand sind zu erwarten – in der Anfangsphase mehr als in der Endphase. Die Abweichungen sollten über den Verlauf des Projektes hinweg geringer werden.
6. **Wer** (Entwickler) hat die Verantwortlichkeit für die User Story **wann** übernommen und wann wurde die User Story abgeschlossen/**abgenommen** vom Auftraggeber.

Weiterhin kann die Rolle der Benutzer erfasst werden, für die die entsprechende User Story von besonderer Bedeutung ist.

Im Allgemeinen bedarf die Erfassung von User Stories keiner besonderen Werkzeuge und die Verwendung von – zum Beispiel – Google Docs ist ausreichend. Es gibt jedoch im Internet auch (freie) Werkzeuge, die verwendet werden können (zum Beispiel: Redmine - <http://www.redmine.org/> bzw. <https://scm.rbg.informatik.tu-darmstadt.de/>).

Am Ende des Projektes sind alle User Stories abzugeben.

Unified Process

Pflichtenheft Aufbau

Das Pflichtenheft muss die folgenden Kapitel enthalten:

- Ist Analyse
- Soll Analyse
- Anforderungen (funktional und nicht-funktional)
- Grobarchitektur
- Qualitätssicherung
(Dieser Abschnitt soll analog zum vorher beschriebenen Qualitätssicherungsdokument ausgeführt werden - auf eine Einleitung zum Projekt ist jedoch zu verzichten.)
- Risikomanagement

- Rechtliches
(Insbesondere in Hinblick auf die Verwertung der erstellten Software.)

Abgabe(n)

Die Vorabgabe für das Pflichtenheft ist ca. 6-8 Wochen nach dem Start des Projektes fällig. Der genaue Termin wird durch den Teamleiter – nach Rücksprache mit der Projektbegleitung – bekannt gegeben. Die Vorabgabe ist optional und soll den Gruppen ermöglichen detailliertes Feedback zu bekommen.

Die Endabgabe erfolgt grob mit dem Ende der Vorlesungszeit. Der genaue Termin wird auch in diesem Fall durch den Teamleiter – nach Rücksprache mit der Projektbegleitung – bekannt gegeben.

Am Endes des Projektes ist darüber hinaus ein Nachweis über die durchgeführten Qualitätssicherungsmaßnahmen zu erbringen sowie kurz zu dokumentieren welche Anforderungen umgesetzt wurden bzw. welche Änderungen sich ergeben haben.

Formal

Es ist die selbe Vorlage zu verwenden, die auch für das Qualitätssicherungsdokument zu verwenden ist. Der Seitenumfang für das Pflichtenheft als Ganzes ist nicht beschränkt.

Teamleiterpraktikum



In diesem Kapitel werden
kurz die wesentlichen
Aufgaben und
Verantwortlichkeiten der
Teamleiter beschrieben

Aufgaben der Teamleiter

Sicherstellung des Erfolgs der betreuten Projekte

Das primäre Hauptziel der Teamleiter ist es sicher zu stellen, dass alle betreuten Gruppen das Projektziel erreichen. D.h. dass alle Auftraggeber am Ende ein Produkt erhalten, dass die gestellten Anforderungen weitgehend erfüllt.

Daraus ergibt sich, dass der Teamleiter in direkter Verantwortung ist folgende Probleme unmittelbar der Projektbegleitung mitzuteilen:

- Der Projekterfolg als solches ist (auch nur möglicherweise) in Gefahr.
- Es gibt gruppeninterne Probleme.
- Die Gruppe ist nicht in der Lage die Qualitätsanforderungen an die Dokumente umzusetzen.

- Die Anforderungen des Auftraggebers sind nicht machbar bzw. der Auftraggeber will Einfluss nehmen auf die interne Struktur der Teams.

Darüber hinaus ist der Teamleiter mitverantwortlich für die Wahl und Überwachung des anzuwendenden Softwareentwicklungsprozesses mit ein. Er trägt somit die Verantwortung für die Planung, Einhaltung und Protokollierung des Projektverlaufs

Damit der Teamleiter seine Aufgaben wahr nehmen kann ist es deswegen erforderlich, dass er intensiven Kontakt zu den Gruppen und dem Auftraggeber hat.

Sicherung der Qualität der Qualitätssicherungsdokumente

Es ist die Aufgabe des Teamleiters die Qualität der Qualitätssicherungsdokumente zu sichern. D.h. der Teamleiter hat sich aktiv in die Erstellung des Dokumentes einzubringen und sicherzustellen, dass das Dokument inhaltlich und auch von der Form her exzellent ist.

Die Abgabe der Qualitätssicherungsdokumente (auch der Vorversionen) erfolgt ausschließlich über den Teamleiter an die Projektbegleitung.

Reporting der Zeiten der Gruppen

Alle Teamleiter müssen vor jeder Teamleitersitzung die Zeiten, die die Gruppen seit dem letzten Teamleitermeeting aufgewendet haben, bei den Gruppen abfragen, um selbige dann der Projektbegleitung zur Verfügung zu stellen.

Folgende Zeiten werden von der Projektbegleitung abgefragt:

- Zeit, die die Gruppe für die Bearbeitung der User Stories bzw. Anforderungen benötigt hat inkl. "Velocity" der letzten abgeschlossenen Iteration.
Wann immer möglich sollten entstandene Aufwände User Stories/Anforderungen zugeordnet werden.
- Sonstige Zeit, die die Gruppe für projektbezogene Tätigkeiten aufgewendet hat (z.B. Auftraggeberentreffen.)
- Zeitaufwand des Teamleiters

Es ist essentiell, dass **alle Aufwände** erfasst werden bis auf:

- Besuch der Projektbegleitungsvorlesung
- Besuch der Präsentationsschulung durch die HDA
- Zeit, die benötigt wurde für die Erstellung der Präsentation, die im Rahmen der Projektbegleitungsvorlesung zu halten ist



Protokoll Schreiben

Im Rahmen jeder Teamleitersitzung wird ein Teamleiter bestimmt, der ein Protokoll der Sitzung erstellt.

Dieses Protokoll ist innerhalb von max. drei Werktagen an die Projektbegleitung zu senden (Word oder Open Office).

Nach einem informellen Review wird dann das Protokoll an alle Teamleiter verteilt - durch die Projektbegleitung.

Benotung

Die Benotung der Teamleiter ergibt sich aus dem Erfolg in der Leitung der Teams. Dies spiegelt sich unter anderem im Projekt-erfolg (gemäß Auftraggeber) und in der Qualität der abgegebe-nen Dokumente wieder.

Informationen für Auftraggeber



Im Folgenden werden kurz die für Auftraggeber wesentlichen Dinge dargestellt.

Anforderungen

Ressourcen

Hardware, Software oder anderweitiges Material, die ein Projektteam ggf. zur Durchführung benötigt, sind von Seiten des Auftraggebers zur Verfügung zu stellen. Das Selbe gilt für etwaige benötigte Räume.

Von Seiten der Projektbegleitung bzw. von Seiten des Fachbereichs wird nur ein Server zur Verfügung gestellt, auf dem ein Versionsverwaltungssystem und eine Projektmanagementsoftware installiert ist.

Verwertung der Software

Bei dem Bachelor-Praktikum handelt es sich um ein Pflichtpraktikum. Es gelten deswegen in Hinblick auf die wissenschaftliche bzw. kommerzielle Verwertung die selben Regeln, wie sie auch im Falle von Praktika etc. gelten. Soll die erstellte Arbeit nach Abschluss in einer bestimmten Art und Weise wissenschaftlich bzw. kommerziell verwertet werden, so ist dies bei der Themen-

vorstellung zu erwähnen und im Rahmen des ersten Treffens noch einmal unmittelbar mit den Gruppen abzustimmen. Von Seiten der Projektbegleitung aus gibt es keine Vorgaben. Der Projektbegleitung muss jedoch ggf. Einsicht in den Code gewährt werden, wenn es zu Problemen im Rahmen der Notenfindung kommt.

Anhang

Weitere Informationen:

<https://www.informatik.tu-darmstadt.de/de/studierende/studiengaenge/bachelor-informatik/bachelor-praktikum/>

Literatur

K. Beck; eXtreme Programming explained - embrace change;
Addison Wesley, 2000

K. Beck und M. Fowler; Planning Extreme Programming; Addison Wesley, 2001

H. Wolf, S. Rook und M. Lippert; eXtreme Programming, 2. Auflage; dpunkt.verlag, 2005

Historie

| Datum | Bearbeiter | Kommentare |
|-------------|-------------|--|
| 11.09.2010 | M. Eichberg | erste Version |
| 30.09.2010 | M. Eichberg | Einarbeitung der Kommentare von Felix Kerger und Guido Rößling |
| 08.10.2010 | M. Eichberg | Einarbeitung von Kommentaren von Wolfgang Heenes |
| 25.11.2010 | M. Eichberg | Ergänzungen im Bereich User Stories (basierend auf Anregungen von Enno Deutschmann und Andreas Kaluza) |
| 11.03.2011 | M. Eichberg | Der Abschnitt über das QS Dokument wurde präzisiert. |
| 18.04.2011 | M. Eichberg | Abschnitt über die Benotung hinzugefügt. |
| 17.05.2011 | M. Eichberg | „allgemeine Überarbeitung“ - insbesondere im Hinblick auf die Ausgestaltung des QS Dokuments |
| 01.08.2011 | M. Eichberg | Hinweis bzgl. Gestaltung des Deckblatts des QS Dokuments hinzugefügt |
| 17.4.2012 | M. Eichberg | Einarbeitung der Erkenntnisse des WS11/12 |
| 10./11.2012 | M. Eichberg | Komplette Überarbeitung des Dokuments (Informationen zum Teamleiterpraktikum und Bachelor-Praktikum zusammengeführt) (Besonderer Dank gilt allen Studierenden, die auf Fehler bzw. Unklarheiten im Dokument hingewiesen haben!) |
| Dez. 2012 | M. Eichberg | Erste grundlegende Informationen über das Pflichenheft hinzugefügt |
| Feb. 2012 | Alle | Informationen über den QS Anhang |
| Mai 2014 | M. Eichberg | Aktualisierung für Sommersemester 2014 |
| April 2016 | M. Eichberg | Aktualisierung für Sommersemester 2016 / Anpassung an neue PO |

Iteration

Fixer, kurzer Zeitraum (eine bis max. 3 Wochen), in der von der Gruppe ausgewählte User Stories implementiert werden.

Related Glossary Terms

Planning Meeting, Retrospektive

Index

[Find Term](#)

Planning Meeting

Treffen zwischen dem Team und dem Auftraggeber, in dem die umzusetzenden Anforderungen der nächsten Iteration besprochen werden.

Related Glossary Terms

Iteration

Index

[Find Term](#)

Retrospektive

Kurzes Meeting des Teams am Ende einer Iteration (eines Releases), um zu besprechen, was gut lief und was verbesserungsfähig ist. Die Retrospektive findet ggf. mit Teamleiter, aber immer ohne Auftraggeber statt.

Related Glossary Terms

Iteration

Index

[Find Term](#)

User Story

Ein kurzer Text, der eine wesentliche (fachliche) Anforderung aus Sicht des Kunden erfasst. Diese Anforderung muss testbar sein und muss sich innerhalb einer Iteration umsetzen lassen. Ggf. ist die User Story aufzubrechen in kleinere Teile. Wenn möglich sollte pro User Story ein Akzeptanztest definiert sein.

Related Glossary Terms

Drag related terms here

Index

[Find Term](#)

Chapter 1 - Agiler Softwareentwicklungsprozess
Chapter 1 - Agiler Softwareentwicklungsprozess